Playing Games with Optimal Competitive Scheduling

Jeremy Frank and **James Crawford** and **Lina Khatib** * and **Ronen Brafman** †
Computational Sciences Division
NASA Ames Research Center, MS 269-3
frank@email.arc.nasa.gov
Moffett Field, CA 94035

Introduction

This paper is concerned with the problem of allocating a unit capacity resource to multiple users within a pre-defined time period. The resource is indivisible, so that at most one user can use it at each time instance. However, different users may use it at different times. The users have *independent, selfish* preferences for *when* and for *how long* they are allocated this resource. Thus, they value different resource access durations differently, and they value different time slots differently. We seek an optimal allocation schedule for this resource.

This problem arises in many institutional settings where, e.g., different departments, agencies, or personal, compete for a single resource. We are particulary motivated by the problem of scheduling NASA's Deep Space Satelite Network (DSN) among different users within NASA. Access to DSN is needed for transmitting data from various space missions to Earth. Each mission has different needs for DSN time, depending on satelite and planetary orbits. Typically, the DSN is over-subscribed, in that not all missions will be allocated as much time as they want. This leads to various inefficiencies – missions spend much time and resource lobbying for their time, often exagerating their needs. NASA, on the other hand, would like to make optimal use of this resource, ensuring that the good for NASA is maximized. This raises the thorny problem of how to measure the utility to NASA of each allocation.

In the typical case, it is difficult for the central agency, NASA in our case, to assess the value of each interval to each user – this is really only known to the users who really understand their needs. Thus, our problem is more precisely formulated as follows: find an allocation schedule for the resource that maximizes the sum of users preferences, when the preference values are private information of the users. We bypass this problem by making the assumptions that one can assign money to customers. This assumption is reasonable; a committee is usually in charge of deciding the priority of each mission competing for access to the DSN within a time period while scheduling. Instead, we can assume that the committee assigns a budget to each mission. We then assume that customers express preferences by attaching a monetary value to each allocation of an interval, and that the utility to NASA is linear in the sum of user values.

Given the users' valuations over different time allocations, the problem is a possibly challenging optimization problem. However, as we noted, these valuations are private information known only to the users, and *a-priori* we have no reason to assume that they will describe it truthfully – in practice missions tend to exagerate these values in order to obtain more DSN time. This problem is reminiscent of similar competitive allocation problems. Many such problems are solved using auction-based mechanisms. The nature of the mechanism and its properties differ depending on the type of good being auctioned and the optimality criteria used. In particular, the structural properties of the goods being auctioned plays an important role in determining the complexity of solving it. Such properties include: the number of goods of each type, whether goods are divisible or not, etc. Although time is a continuous property, it is not a classical divisible good because different allocations with identical durations can be valued differently. Moreover, because of their time constraints certain tasks may simply be mutually exclusive. It is also not a standard indivisible good because we do have some flexibility with the length and starting time of tasks.

To the best of our knowledge, preference elicitation for scheduling with this wide range of features has not been addressed in the literature.

The main contributions of this paper are as follows:

- We formally define the Optimal Competitive Scheduling (OCS) problem for Deep Space Network (DSN). The problem combines preference elicitation, bidding and clearing of an auction for a constrained resource. The problem differs from typical optimal competitive resource allocation problems due to the mix of continuous good allocations with discrete (combinatorial) allocations.
- We describe a novel incentive compatible VCG mechanism for this problem, i.e. the mechanism ensures that it is a (weakly) dominant strategy for the agent's to reveal their true preferences.
- We show that the general version of OCS is $\mathcal{NP}$-complete, and we identify an interesting restricted subclass for which a polynomial time mechanism exists. This

*Kestrel
†QSS

subclass generalizes the class of Simple Temporal Problems with Preferences (STPPs).

- We provide a preliminary empirical study of our algorithm on randomly generated tractable cases of OCS problems.

The paper is organized as follows. In Section 2 we review the relevant related results. In Section 3 we formally define the OCS problem. In Section 4 we discssu a VCG-like mechanism for OCS. In Section 5 we describe a tractable class of OCS. Preliminary empirical results are presented in Section 6. We conclude in Section 7.

Definitions

We recall a number of needed definitions and previous related work. We begin with some standard definitions from the literature on auctions. We assume the auction is for a unit resource over a fixed time horizon. There is one seller and N buyers. We restrict our attention to one-shot sealed bid auctions with no reserve price.

The first definition describes a basic auction mechanism:

Definition 1 *A* direct revelation mechanism *receives as input a vector of valuations and produces as output an allocation and a vector of payments where each bidder receives their allocation and pays their payment.*

In general, agents' valuations need not necessarily reflect their true preferences. An incentive compatible mechanism is one where agents have no advantage misrepresenting their preferences.

Definition 2 *A direct revelation mechanism is* incentive compatible *if for every bidder A, every true valuation v_A, all declarations of other bidders v_{-A} and all possible false declarations v_A^f, the utility of biding bidding v_A^f by agent A under the mechanism does not exceed agent A's utility bidding the true valuation v_A. That is, let v_A be A's valuation, λ_A be A's allocation and P_A be A's payment when A bids truthfully, is allocated λ_A, and the other agents bid v_{-A}. Let λ_A^f and P_A^f be A's allocation and payment when A bids v_A^f, is allocated λ_A^f, and the other agents bid v_{-A}. Then, we must have that $v_A(\lambda_A) - P_A \geq v_A(\lambda_A^f) - P_A^f$.*

The best known technique for generating a direct revelation mechanism is via the Vickery-Clarke-Groves mechanism. It can be viewed as a generalization of second price auctions.

Definition 3 *Let v be a valuation vector for N agents. The Vickery-Clarke-Groves (VCG) mechanism computes the allocation with maximal sum of valuations. Let λ_A be agent A's allocation. Agent A's price is set to the maximal sum of valuations assuming agent A's valuation was altered to reflect a value of 0 for λ_A.*

VCG mechanisms are designed to minimize the impact of the "winners' curse". While bidding, if A theorized winning an auction and paying the bidding price, then A would lower its bid on the assumption that it bid too much. The VCG mechanism forces A to reason about the prices all the other bidders might make; A is now forced to bid his true valuation function.

Next, we recall the definition of a tractable class of optimal scheduling problems: Simple Temporal Problems with Preferences (STPPs) ().

Definition 4 *An STPP consists of a set of events E, a set of simple temporal constraints of the form $a \leq |e_i - e_j| \leq b$ where $e_i, e_j \in E$, and a set of real-valued preference functions on some event distances: $f(|e_i, e_j|)$. One event, e_0, is designated as the origin (i.e., $e_0 = 0$), and constraints involving e_i and e_0 are interpreted as unary bounds on when e_i can occur. The problem is to find a feasible assignment to $e_i (\neq e_0)$ i.e., an assignment satisfying the constraints maximizing $\sum_{e_i,e_j} f(|e_i - e_j|)$ that also satisfies all constraints.*

Previous Work

Our problem has two central aspects: inducing the agents to reveal their true preferences and solving the associated optimal scheduling problem. Previous work roughly falls into these two categories.

Infinitely divisible goods auctions are usually found in treasury market settings. (BZ93) consider the optimization of auctioneer's revenue in a shared-value auction of an (almost) infinitely divisible good, namely treasury bills. This paper is principally concerned with comparing VCG mechanisms (called uniform price auctions) to first-price auctions (called discriminatory) price auctions. While intriguing, the work does not seem to shed light on how to construct mechanisms for our OCS problem.

Some authors have described incentive compatible mechanisms for multi-unit combinatorial auctions. (BGN02) describe a tractable, incentive compatible mechanism for a multi-unit combinatorial auction that does not maximize the seller's profits. The auction assumes monotonicity in bidders' demands, that is, the more of a good they receive the more they are willing to pay. This approach may not scale to our full problem, and also does not reflect the hybrid nature (mixes of discrete and continuous decisions) in our setting. (Ms03) describe decentralized maximize weighted social welfare auctions. Although interesting, distributed mechanisms are not an issue in our case. They also explore problems like lying auctioneer that we do not worry about.

Finally, some authors have considered simple scheduling auction settings. (Rou04) considers game-theoretic scheduling, but uses only simple scheduling models in which the value of tasks is derived from the load on the machines on which they are scheduled. This can again be considered a shared-value setting, as the value to the bidder is a function of not only their award, but the award of other bidders. Furthermore, there is no task selection and no task ordering component to the problem considered here. (Ms03) also discuss scheduling problems of this nature, again using a decentralized auction setting to solve the problem. (SS00) show that very restrictive forms of our problem have polynomial time algorithms; these require tasks with fixed duration, and hence fixed preference value, and no flexibility in start times. While relevant and possibly generalizable, the setting is far simpler than our general model.

The literature on scheduling contains vast numbers of tractable cases for scheduling problems with one optimiza-

tion criteria. The work most related to our type of optimization problem is that on STPPs. In (MMK$^+$04) it was shown that if the preference functions over $|e_i - e_j|$ are restricted to be piecewise linear and convex, the STPP can be solved by formulating an appropriate linear program (LP), rendering its solution time polynomial. It is important to note that in an STPP, all events must be assigned a positive time. If we view events as task start and end times, this means that in order to formulate a scheduling problem as an STPP, we must first resolve all resource conflicts either by rejecting certain tasks or by ordering tasks. Thus, in practice the input consists of a set of ordered tasks, all of which must be allocated some time. Cases where there are too many tasks or sets of tasks have trivially infeasible constraints cannot be resolved by the existing techniques.

The Optimal Competitive Scheduling Problem for the Deep Space Network

We begin with a resource R, a time horizon $[0..h]$, and a set of bidders J. Let $\mathcal{A}$ be a set of activities. Let S_A be the start time, D_A be the duration, and E_A be the end time of activity $A \in \mathcal{A}$. Associated with A are the following constraints: $S_A + D_A = E_A$, All activities A also share a single unary resource R (i.e. jobs can't overlap). We may also associate other constraints among collections of activities. The most elementary such constraints are absolute start and end time: $0 \leq s_{A_{lb}} \leq S_A \leq s_{A_{ub}} \leq h$, and $0 \leq e_{A_{lb}} \leq E_A \leq e_{A_{ub}} \leq h$. Similarly, tasks can have minimum and maximum durations: $0 \leq d_{A_{lb}} \leq D_A \leq d_{A_{ub}} \leq h$. Another simple form of constraint are Simple Temporal Constraints (DMP91) of the form $a \leq T_A - T_B \leq b$, where $T_A = \{S_A, E_A\}$, which can enforce activity orderings and activity endpoint separations.

Bidders express preferences over *when* individual activities take place, *how much* activities are separated, and *how long* activities last. For example, a common preference is that longer duration activities are better. Another common preference is for activities to start at some time t. In the DSN setting, the first of these preferences is the most common. For simplicity, we assume that each bidder bids on one task; in what follows, tasks and bidders will be referred to identically[1]. Thus, players' preferences are functions of the form $v_A(S_A, D_A)$. This model assumes that the activities and constraints between activities are known to bidders ahead of time. In order to make it possible for players to bid, we imagine the following protocol: 1) tasks published 2) money distributed 3) players submit bids 4) mechanism solves for schedule 5) prices paid.

The auction protocol is a single sealed-bid auction; players bid, winners and prices are determined, and winners determine the schedule that will be executed. Thus, the winning bids must collectively define feasible schedules. However, if there is a constraint mentioning A but A is rejected from the final schedule, the constraint is meaningless and need not hold in the final schedule.

[1]Much of our development generalizes to scenarios with multiple bids per task.

The Optimal Competitive Scheduling Problem (OCS) is to maximize the sum of the value of players' true preferences over schedules given the constraints and bids by choosing a subset of bids $\mathcal{A}'$ to satisfy and choosing the values of $S_A, E_A, D_A \forall A \in \mathcal{A}'$ satisfying the relevant constraints (including the resource constraints).

Problem Complexity

Computing an optimal allocation for the agents' declared preference functions is a necessary step for a VCG-like mechanism. In general, any mechanism will produce, as a by-product, a solution to the optimal allocation problem with respect to the agents' true preference functions. OCS is an extension of $1|r_i; p_i; d_i| \sum_i w_i U_i$ (Brü98); that is, scheduling tasks with release times, due dates, and duration constraints on a unary resource in order to minimize the weighted penalty of missed jobs (alternatively maximize value of completed jobs) with duration-dependent values for tasks, additional opportunities for positive value (start time preferences), and additional constraints (simple temporal constraints on timepoints). Karp has shown that $1|r_i; p_i; d_i| \sum_i w_i U_i$ is $\mathcal{NP}$-complete; thus, OCS must be $\mathcal{NP}$-hard.

A Second Price (VCG) Mechanism for OCS

Building a VCG mechanism for OCS is a little tricky. Suppose v_A is only a function of the duration of task A, i.e. $v_A(D_A)$. Let $D_A = \lambda_A$ be A's allocation and P_A be A's payment. Suppose λ_A are the allocations maximizing $P = \sum_{A \in \mathcal{A}} v_A(\lambda_A)$. Suppose we only eliminate the winning allocation λ_A by enforcing $v_A(\lambda_A) = 0$. Then to find the second price we can build 2 optimization problems: the first problem is identical to the original optimization problem with the added constraint $D_A < \lambda_A$ and the second is identical to the first with the added constraint $D_A > \lambda_A$; let the maxima for these two problems be $P^<, P^>$ respectively. The second price is $max((P^> - (P - v(\lambda_A)), (P^< - (P - v(\lambda_A)))$. But since we impose $<$ constraints it's clear that the second price is virtually identical to the first price, implying $P_A = v(D_A) \pm \epsilon$! In general, if v_A depends on k variables, 2^k optimization problems are needed to justify the second price, but the "proof" suffices to show that 2d price always equal to first price. Such a mechanism is essentially not a VCG mechanism, because the bidders can *prove* to themselves that the second price always equals the first price. This means that, once again, bidders will be able to reason about the price they will pay as a function of their own bid, and have incentive to under-bid.

Mechanism and Proof of Complexity

To help with creating our mechanism we repeat Theorem 1 of (BGN02):

Theorem 1 *A direct revelation mechanism is incentive compatible if and only if, for every bidder A and every vector of bids v_{-A}: 1) for every allocation λ, $P_A(\lambda)$ is not a function of v_A, and 2) for each v_A, λ_A maximizes $v_A(\lambda_A) - P_A(\lambda_A)$ for all legal allocations λ_A.*

Our problem in creating the VCG mechanism above was that we only eliminate a set of measure zero from an effectively infinite space of bids. The result is that there is provably no detectable shift in the price; alternately, there is too much information about v_A entering the price calculation. One way to repair this is to force v_A's bid for *any* value of D_A to zero. More generally, rather than eliminate A's bid for a particular assignment, we can simply eliminate A's bid entirely. If P^{-A} is the value of the optimal solution without $A's$ bid, then the second price is $P^{-A} - (P - v(\lambda_A))$.

Theorem 2 *The above mechanism is incentive compatible.*

We will show our allocation satisfied the conditions of Theorem 1 of (BGN02). The value $P - v(\lambda_A)$ is the value of the allocation λ_{-A} to all other bidders. The value of this allocation will be the same in the allocation defining P^{-A}. If it weren't then P would not maximize $\sum_{A \in \mathcal{A}} v_A(\lambda_A)$. This can be shown by recasting the auction as a multi-good combinatorial auction, and analyzing the mechanism as applied to the new auction, but space precludes the full exposition. The value of the allocation to A redistributed to $B \neq A$ depends only on v_{-A}; thus, $P^{-A} - P + v(\lambda_A)$ only depends on $v - A$, satisfying Condition 1. Now suppose v_A is fixed. The mechanism awards λ_A to A if its bid *maximally* exceeds that of other bids for the same allocation. That means any other allocation λ'_A with $v_A(\lambda'_A) > v_A(\lambda_A)$ has the property that $P_A(\lambda'_A) > v_A(\lambda'_A)$. This satisfies Condition 2.

The Tractable Case: Extending STPPs

We showed earlier that the OCS problem is $\mathcal{NP}$-hard. Thus, in general, we would expect to use branch-and-bound search to find an optimal allocation. In this paper, we identify and solve a tractable case of OCS. In the future, we would expect to use such tractable cases to produce bounds for the general case.

We restrict the temporal constraints to be activity ordering constraints (i.e., constraints on the relation between the start times of different tasks). For simplicity, we will continue to refer to these constraints using the same notation previously used for simple temporal constraints: $(a \leq |T_A - T_B| \leq b)$. We will also impose limits on the form $v_A(D_A)$ can take. We first assume every function $v_A(D_A)$ is a set of k_A piecewise linear functions, $\{f^k_{D_A}\}$, where $f^k_{D_A}$ is defined over an interval $[x_{k-1}, x_k]$ and $f^k_{D_A}(x_k) = f^{k+1}_{D_A}(x_k)$. Furthermore we assume each $v_A(D_A)$ is convex.

Given the above restricted version of OCS, suppose that we have predetermined the relative order of all conflicting tasks. Then, we can use the LP formulation of the STPP to solve for the task durations. The LP contains a variable representing the start, duration and end of each task (S_A, D_A, E_A). We also introduce a new variable V_A. The form of the LP is as follows: To build the LP formulation of the STPP, we have:

maximize $\sum v_A$
subject to: forall A
$s_A + D_A = E_A$
$s^l_A \leq S_A \leq s^u_A$,
$e^l_A \leq E_A \leq e^u_A$,
$d^l_A \leq D_A \leq d^u_A$,
$a \leq |T_A - T_B| \leq b$
$\forall k_A \; V_A \leq f^{k_A}_{D_A}(D_A)$

We refer the reader to (MMK+04) for the proof that the solutions to this LP are the optimal solutions to the STPP.

Extending the STPP

The requirement that all conflicting tasks be ordered is not particularly natural in our case. Moreover, even then, if there are too many tasks, the system of linear constraints in the LP may be trivially infeasible. However, if it just so happens that all activities that could overlap on the resource are ordered, and all tasks allow for the possibility of zero duration then the STPP is trivially feasible, and "maximal" (no subset of the STPP has a better optimal solution).

The requirement that we be able to reduce the duration of a task to zero allows the LP algorithm to reject tasks without conducting combinatorial search. However, we must still order the tasks. We now identify a restricted set of problems for which the best possible ordering can be found in polynomial time. From the above discussion it follows that for such problems the optimization problem is polynomial (assuming zero-duration is allowed).

Definition 5 *Two tasks A, B are a* containment pair *if $s_{A_{lb}} \leq s_{B_{lb}}$ and $e_{B_{lb}} \leq e_{A_{lb}}$.*

Theorem 3 *Given an OCS problem whose preferences are piecewise linear convex functions over D_A, and whose metric temporal constraints are trivially feasible and limited to time windows and task ordering. If no pair of tasks A, B is a containment pair, then the ordering according to $s_{A_{lb}}$ induces an STPP whose optimal solution is maximal over the STPPs induced by any other task ordering.*

In order to prove this we prove that in any feasible ordering where A, B are not ordered according to this rule we can re-order A, B and eliminate fewer choices of task duration due to consistency enforcement. Since the optimization criteria of the LP is based on task duration, this will prove the result. Figure 1 shows the last three cases pictorally. Assume w.l.o.g. $s_{A_{lb}} \leq s_{B_{lb}}$ and $e_{A_{lb}} \leq e_{B_{lb}}$, thus the proper ordering should be $A \leq B$. There are 4 cases;

- Suppose A, B are mis-ordered and there are no tasks C such that $B \leq C \leq A$. The ordering $B \leq A$ prunes the windows of A and B to $[s_{B_{lb}}, e_{A_{ub}}]$, thus (possibly) reducing the feasible durations. Re-ordering to $A \leq B$ leads to no pruning of start times or durations.
- Now suppose there is at least one task C such that $B \leq C \leq A$ with $s_{C_{lb}} \leq s_{A_{lb}}$ and $e_{C_{lb}} \leq e_{A_{lb}}$. The constraint $B \leq C$ forces all tasks' windows to be $[s_{B_{lb}}, e_{C_{ub}}]$; reordering to $A \leq C \leq B$ loosens the windows to $[s_{A_{lb}}, e_{C_{ub}}]$.
- Now suppose there is at least one task C such that $B \leq C \leq A$ with $s_{A_{lb}} \leq s_{C_{lb}}$ and $e_{A_{lb}} \leq e_{C_{lb}}$, and $s_{C_{lb}} \leq s_{B_{lb}}$ and $e_{C_{lb}} \leq e_{B_{lb}}$. In this case the constraint $B \leq A$ is the tightest constraint, forcing the windows of all tasks to be $[s_{B_{lb}}, e_{A_{ub}}]$; reordering $A \leq B$ imposes no pruning of start times or durations.

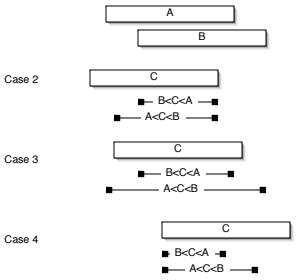

Figure 1: The last 3 cases of the optimal ordering proof.

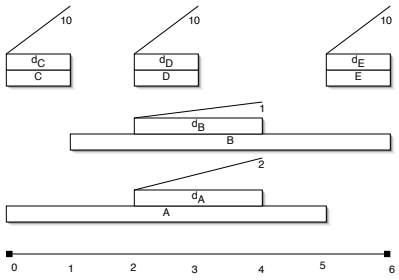

Figure 2: The case where containment pairs induce reversals of the optimal ordering for non-containment pair tasks.

- Now suppose there is at least one task C such that $B \leq C \leq A$ with $s_{B_{lb}} \leq s_{C_{lb}}$ and $e_{B_{lb}} \leq e_{C_{lb}}$. In this case the constraint $C \leq A$ forces all windows for tasks to be $[s_{C_{lb}}, e_{A_{ub}}]$; reordering to $A \leq C \leq B$ loosens the windows to $[s_{C_{lb}}, e_{B_{ub}}]$.

Now suppose we have a containment pair with A containing B. If we assume tasks are *interruptible* we can split A into A' and A'' such that the optimal ordering is $A' < B < A''$ according to the previous rule. Initially, this may not appear to be an equivallent problem since our preference for the duration of A'' depends on the duration of A'. However, we can use the sum of their durations directly in the objective function for the induced LP.

It is tempting to think that we can find the optimal ordering for a containment pair without interruptibility. Either ordering prunes duration choices for A but neither prune duration choices for B. It is trivial to determine which prunes more duration choices for A (we can construct cases for which either ordering prunes more choices). One might think that the optimal ordering is that which prunes fewer duration choices for A. Unfortunately, not only is this not true, but we can demonstrate that containment pairs actually break the "optimal" ordering for non-containment pairs of tasks. The following simple example (also shown in Figure 2) shows this:
Task A: $s_{A_{lb}} = 0, e_{A_{ub}} = 5, d_{A_{ub}} = 2, v_A(D_A) = 2d_A$
Task B: $s_{B_{lb}} = 1, e_{B_{ub}} = 6, d_{B_{ub}} = 2, v_B(D_B) = d_B$
Task C: $s_{C_{lb}} = 0, e_{C_{ub}} = 1, d_{C_{ub}} = 1, v_C(D_C) = 10d_C$
Task D: $s_{D_{lb}} = 2, e_{D_{ub}} = 3, d_{D_{ub}} = 1, v_D(D_D) = 10d_D$
Task E: $s_{E_{lb}} = 5, e_{E_{ub}} = 6, d_{E_{ub}} = 1, v_E(D_E) = 10d_E$

The idea is that highest paying tasks (C, D,and E) should be placed in the schedule and given their maximum duration. This leaves 2 gaps to be filled by lower paying tasks (A and B). The gaps are as follows: [1 2] and [3 5]. If $A < B$, A gets assigned [1 2] and B gets assigned [3 5]. The value is 2+2=4. If $B < A$, B gets assigned 1 slot [1 2] and A gets assigned 2 slots [3 5]. The value is 1+4=5 Thus, $B < A$ is the optimal ordering for A and B, contradicting the optimal ordering when no containment pairs are present.

generateProblem(h,w,d,v,p,n)

for n tasks
  Choose window size w_n u.a.r. from [1..w]
  Choose start time u.a.r. from $[1..h - w_n]$
  while task window *contains* another task window
    Shrink window; update w_n
  while task window *contained by* another task window
    Enlarge window; update w_n
  Choose duration d_n u.a.r. from $[1..\min(d, w_n)]$
  Choose maximum bid b_n u.a.r. from $[1..v]$
  Choose number of pieces of preference u.a.r. from $[1..p]$
  for each piece of preference
    #$d*$ is remaining duration w_n, $b*$ is remaining bid height b_n
    Choose piece duration d_i u.a.r. from $[1..d*]$; update d*
    Choose bid height b_i u.a.r. from $[1..b*]$; update b*
    Add preference function piece of duration d_i up to bid b_i

Empirical Results

In this section we describe some preliminary empirical results on a VCG mechanism applied to randomly generated instances of these problems.

Random Problems

The instances are generated as follows. Let h be the scheduling horizon. Let w be the maximum window for any task, d be the maximum duration of any task, v be the maximum bid for any task, and p be the maximum number of pieces of the preference function, also inputs. First, the task start time and maximum duration are selected uniformly from the given range. These characteristics are then modified to ensure that no pair of tasks is a containment pair. Finally, the maximum bid of the task is generated, a random number of pieces of the preference function is selected, and the preference function is generated.

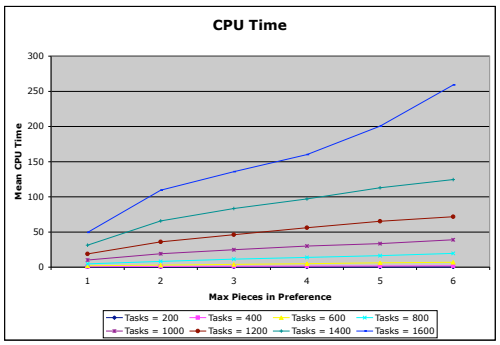

Figure 3: CPU times for randomly generated problems with $h = 4000, w = 20, d = 15$ and $v = 40$.

Preliminary Empirical Results

We present the results of preliminary empirical tests designed to show the growth in time to solve the LP as a function of the problem [2]. Due to the relative simplicity of the temporal constraints on tasks, the principal contributors to the LP are the number of tasks and the number of pieces in tasks' preference functions. Figure 3 shows timing results for problems with $h = 4000, w = 20, d = 15$ and $v = 40$. We vary the number of tasks n from 200 to 1600 by 200, and the maximum number of pieces of the bid p from 1 to 6. Growth in problem complexity is sub-linear in p, as expected, and appears polynomial in the number of tasks. The CPU times shown are in seconds; we average over 5 randomly generated problems. Timing is only for solving the LP, which was done by lpsolve, a public domain LP solver.

Conclusions and Future Work

We describe the OCS problem for Deep Space Network. This problem consists of activities with time windows and flexible durations, simple temporal constraints, and preferences over a combination of start time and duration of tasks. We have shown that a VCG mechanism exists for this problem, which (to our knowledge) is the first mechanism for an auction on a mixture of divisible and indivisible goods. We have identified a class of OCS problems with a tractable second-price mechanism: activities with time windows, flexible duration, ordering constraints, convex, piecewise linear preferences, no containment pairs, and for which duration of zero is feasible. The tractability result is a novel extension of the theory of STPPs to tractably handle task ordering and task rejection.

Obviously a study of the general case of OCS for the DSN is worthwhile. Task containment, metric temporal constraints, multi-capacity resources and choices over resources will generally lead to $\mathcal{NP}$-hard problems. We are investigating options for incorporating the solving of tractable subcases in a complete solver, in a manner similar to that described in (WS05). The tractable cases for STPP described in this paper provide methods for bounding above the value of schedules by relaxing the STPP constraints; they can then be used both as inference mechanisms and as the bases of heuristics. Proving the value of the bounds and subsequently experimenting with those bounds inside a complete solver are on our agenda.

In our formulation, we implicitly assumed that the facility owner (NASA) was the auctioneer and thus had no bid. However, we could generally introduce new bidders to represent the facility owner's criteria on schedules. For example, maximizing utilization could be a preference; NASA could be given a budget for this, and express preferences over it, thereby influencing the auction. In general, we believe the tractability results will extend to multiple preferences over a set of activities.

The tractable cases of OCS for DSN restrict the nature of the preference functions to piecewise linear convex functions of the duration, $v_A(D_A)$ and forces preferences to extend to $D_A = 0$. If more general preference functions $v_A(D_A, S_A)$ are allowed, care must be taken to ensure that rejected tasks derive no value, or formally, $v_A(0, S_A) = 0$. Furthermore, metric temporal constraints on rejected tasks should not be enforced. The STPP framework may not be appropriate in this setting; we are investigating other formalisms in search of new tractable algorithms.

[2] Investigations of solution quality, task rejection and so on are pending.

References

Y. Bartal, R. Gonen, and N. Nisam. Incentive compatible multi unit combinatorial auctions. In *Proceedings of the Dagstuhl Workshop on Electronic Market Design*, 2002.

P. Brücker. *Scheduling Algorithms*. Springer, 1998.

K. Back and J. Zender. Auctions for divisible goods: On the rationale of the treasury experiment. *The Review of Financial Studies*, 6(4):733 – 764, 1993.

R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–94, 1991.

P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In *Proceedings of the* 10^{th} *International Conference on the Principles and Practices of Constraint Programming*, pages 408 –422, 2004.

R. T. Maheswaran and T. Başar. Nash equilibrium and decentralized negotiation in auctioning divisible resources. *Journal of Group Decisions and Negotiation*, 13(2), 2003.

T. Roughgarden. Stackelberg scheduling strategies. *SIAM Journal of Computing*, 33(2), 2004.

T. Sandholm and S. Suri. Improved algorithms for optimal winner determination in combinatorial auctions and generalizations. In *Proceedings of the* 17^{th} *National Conference on Artificial Intelligence*, pages 90–96, 2000.

X. Wang and S. Smith. Retaining flexibility to maximize quality: When the scheduler has the right to decide activity duration. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2005.